Threadpool architecture

public ThreadPoolExecutor(int corePoolSize,// minimum number of threads  
 int maximumPoolSize, // max num of threads  
 long keepAliveTime,//how long does an idle thread stay on for   
  [TimeUnit](https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/TimeUnit.html) unit, / time unit to keep alive ie seconds   
  [BlockingQueue](https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/BlockingQueue.html)<[Runnable](https://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html)> workQueue)/ how large is the thread pool

Creates a new ThreadPoolExecutor with the given initial parameters and default thread factory and rejected execution handler. It may be more convenient to use one of the [Executors](https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executors.html) factory methods instead of this general purpose constructor.

**Parameters:**

corePoolSize - the number of threads to keep in the pool, even if they are idle, unless allowCoreThreadTimeOut is set

maximumPoolSize - the maximum number of threads to allow in the pool

keepAliveTime - when the number of threads is greater than the core, this is the maximum time that excess idle threads will wait for new tasks before terminating.

unit - the time unit for the keepAliveTime argument

workQueue - the queue to use for holding tasks before they are executed. This queue will hold only the Runnable tasks submitted by the execute method.

We use the Excutor.newFixedThreadPool() class that is a class full of premade threadpools so we dont have to deal with making a threadpool with all the options. There is also Executor.newSingleThreadExecutor()